REMARKS

These remarks are being filed in response to the Office action dated January 27, 2003. All objections and rejections are respectively traversed. Reconsideration and further examination of the application in view of the following remarks are respectfully requested.

§103

At paragraph 4 of the Office action, claims 1-37 were rejected under 35 U.S.C. §103 as being unpatentable over U.S. Patent No. 5,742,812 issued to Baylor et al. (hereafter "Baylor") in view of U.S. Patent No. 5,634,122 issued to Loucks et al. (hereafter "Loucks").

Description of the Present Invention

The present invention is directed to a computer system in which a first process, preferably residing in a server node, maintains a data file. A second process, which may reside at a client node, generates a **single first message** that requests that the first process grant the second process **a plurality of tokens** that are required to modify at least one characteristic of the data file. In response to the request, the first process generates a second message that grants **the plurality of tokens** to the second process, if they are available. If any of the tokens are unavailable, i.e., one or more of the requested tokens were previously granted to a third process, the first process may generate a third message directing the third process to relinquish those tokens. In response, the third process may generate a fourth message relinquishing the identified tokens.

2

Notably, the number of tokens requested in the **single first message** is **a plurality of tokens.** Likewise, the third and fourth messages may each specify a plurality of tokens, namely the tokens whose previous grant is to be revoked and relinquished, respectively.

Description of the Cited References

Baylor describes a technique for guaranteeing atomic execution of access requests to access data that spans multiple file server nodes in a parallel file system (See Col. 2, lines 30-33). In Baylor, a parallel file system contains "N" file server nodes ordered from "0" to "N-1" (See Col. 3, lines 31-34, Col. 4, lines 3-4, and Fig. 1). Data of an individual file may be spread over any number of the server nodes (See Col. 3, lines 34-36, and Col. 4, lines 6-7). A requesting site accesses a file by transmitting an access message to each of the file servers containing portions of the file (See Col. 3, lines 59-65). Alternatively, the requesting site may send an access message to one of the server nodes containing portions of the file and the server node, in turn, relays a message containing request information to the other file server nodes containing portions of the file (See Col. 3 line 65 to Col. 4, line 2).

One of the access messages contains a unique identifier that identifies the message as a primary access component. This message is sent to a primary server node. Another of the access messages contains a unique identifier that identifies the message as an ultimate access component. This message is sent an ultimate server node (See Col. 4, lines 21-42). In response to receiving a primary access message, the primary server node generates a token, which is relayed to the next subsequently higher numbered server node

3

(See Col. 4, lines 63-65; also Col. 5, lines 16-21). The token is forwarded to the next higher server node and so on, until the token reaches the ultimate node, i.e., received by the last server node in the order of nodes to receive an access messages (See Col. 4, lines 34-41, also Col. 5, lines 16-23). Execution of the request portion of the access message received at a particular node is not performed until the token corresponding to the request has been received (See Col. 5, lines 24-29).

In summary, with Baylor, a single token is generated to access data in a file in a parallel file system. The single token is passed only among the server nodes containing portions of the file from a primary server node to an ultimate server node. A node receiving an access message associated with the request does not execute the request until it has received the token.

Loucks describes a technique for controlling access to shared resources in a distributed computer system using a token manager (See Abstract). In order to access (e.g., perform an operation on) the shared resource, a process (e.g., a client process) must hold a single token associated with the operation. The token represents an authorization for the client process to perform the operation (See Abstract, also Col. 6, lines 8-18). If the client process does not hold the token, it requests the token from a token manager (See Col. 6, lines 12-16). The token manager examines a list of tokens it holds and depending on the content of the list, may request a new token from a local token manager, request that another client process revoke a token, or simply grant an available token to the requesting client process (See Col. 6, lines 28-36).

4

When a token is requested from the local token manager, the local token manager determines if the requested token conflicts with other outstanding tokens (See Col. 7, lines 2-7). If no conflict exists, the local token manager grants the token. Otherwise, if a conflict exists (e.g., the token is granted to another process), the local token manager first revokes the token from the process holding the token, and then grants the token to the requesting client process (See Col. 7, lines 6-12). Specific token requests may cause the revocation of multiple tokens. For example, a request for an exclusive write token on a file causes the token manager to revoke all granted read tokens on that file before the exclusive write token is granted to the requesting process (See Col. 7, lines 9-12).

In summary, with Loucks a process requests <u>a single</u> token from a token manager in order to access a shared resource. The token manager, in turn, either grants the token immediately, requests the token from a local token manager, or revokes the token from another process. Moreover if the requested token is not available, depending on the type of the token requested, the token manager may, in turn, revoke several <u>other types of tokens</u> from other processes before granting the <u>requested single token</u>.

Differences between the Present Invention and the Cited References

Representative claim 1 recites in relevant part:

"a first process that maintains a data file in a computer-readable memory",

"a **second process** that generates a **first message requesting** that said second process be granted by said first process <u>a plurality of tokens</u> required **for said second process to modify at least one characteristic of said file**", and

5

"said first process generating a second message . . . that grants said tokens to said second process".

In other words, the present invention recites **a second process that explicitly requests a plurality of tokens that are needed to modify a data file by the second process from a first process via a single message (first message).** The first process supplies the **plurality of requested tokens** to the second process via a second message issued to the second process. Baylor and Loucks, either singly or combined, fail to teach or suggest such an arrangement.

In particular, Baylor teaches a technique where a client process performs an operation on a file distributed across a series of file server nodes by sending messages to each node in the series or, alternatively, to a single server node which relays a message containing request information among the file server nodes. A server node designated as a primary node, generates <u>a single token</u> that is passed to each node in the series from the primary node to an ultimate node. A server node services its portion of the request, when it receives the token, and passes the token to the next node. This continues until the ultimate node receives the token and services its portion of the request. The technique described by Baylor uses a <u>single token</u> to accommodate atomic execution of a request that spans multiple file server nodes. Nowhere does Baylor suggest or teach Applicant's novel "a second process that generates **a first message requesting** that said second process be granted by said first process **a plurality of tokens required for said second process to modify at least one characteristic of said file.**"

6

Likewise, Loucks teaches a technique that involves the use of <u>a single token</u>. The token represents an authorization for a process to perform an operation on a shared resource. Here, the client process issues a single request that it be granted the <u>single token</u> from a token manager before the process performs the operation on the shared resource. The token manager operates in a manner similar to the prior art referred to in the Background section of Applicant's Specification, in that if the token is available, the token manager grants the token to the requesting process immediately; otherwise, the token manager acts to revoke the token from another process first before granting it to the requesting process.

As shown, Loucks fails to provide any teaching or suggestion for a second process **requesting a plurality of tokens, via a single first message** that are needed to modify a data file by the second process. Rather, Loucks describes a technique where <u>a single message</u> (request) is made by a process to request <u>a single token</u> that represents an authorization to perform an operation on a shared resource. Nowhere does Loucks suggest or teach Applicant's novel "a second process that generates a first message requesting that said second process be granted by said first process a plurality of tokens required for said second process to modify at least one characteristic of said file."

Applicant's novel technique is superior over prior techniques, including the techniques described in Loucks and Baylor, in that Applicant's technique requests a plurality of tokens in a single message to modify a file. Further, in accordance with Applicant's technique, revocation and relinquishment of tokens may each be accommodated with single messages containing a plurality of tokens that are to be revoked and relinquished,

7

respectively. Advantageously, the inventive technique permits the amount of network bandwidth consumed by tasks associated with a network file system to be substantially reduced compared with the prior art.
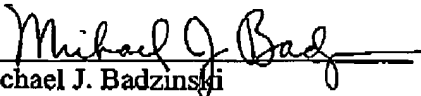
Because both Baylor and Loucks fail to suggest or teach, either singly or in combination, Applicant's novel "a **second process** that generates **a first message requesting** that said second process be granted by said first process **a plurality of tokens required for said second process to modify at least one characteristic of said file,**" Baylor and Loucks are legally precluded from rendering Applicant's claimed invention obvious under U.S.C. 35 §103.

For the reasons set forth above, Applicant submits that all independent claims are believed to be in condition for allowance and that all dependent claims are believed to be dependent from allowable independent claims and therefore in condition for allowance.

Favorable action is respectfully solicited.

Please charge any additional fee occasioned by this paper to our Deposit Account No. 03-1237.

Respectfully submitted,

Michael J. Badzinski
Reg. No. 51,425
CESARI AND MCKENNA, LLP
88 Black Falcon Avenue
Boston, MA 02210-2414
(617) 951-2500

8